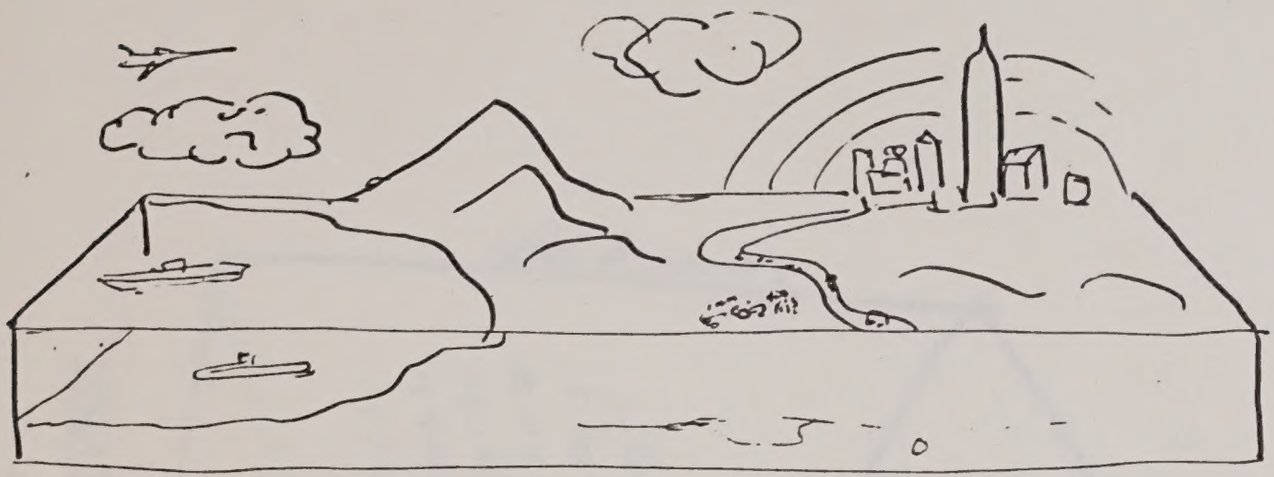
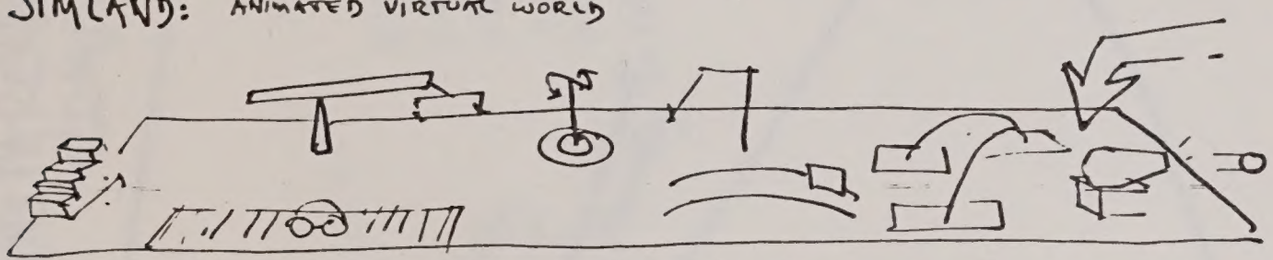
NE CTY

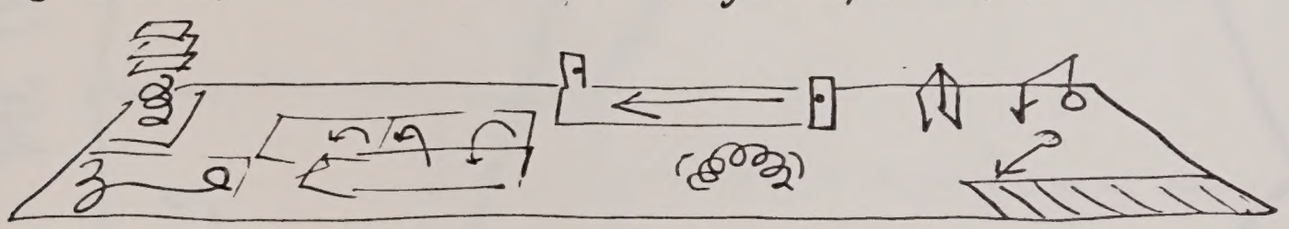
1. THE CARPET OF DATA, building up to Simland.



SIMLAND: ANIMATED VIRTUAL WORLD



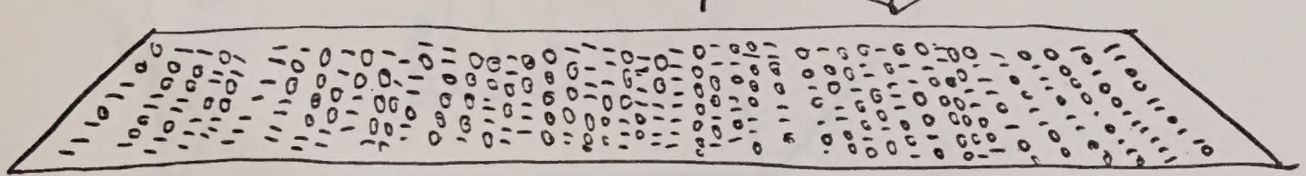
CONSTRUCTS & PROCESSES, subroutines, algorithms, connections



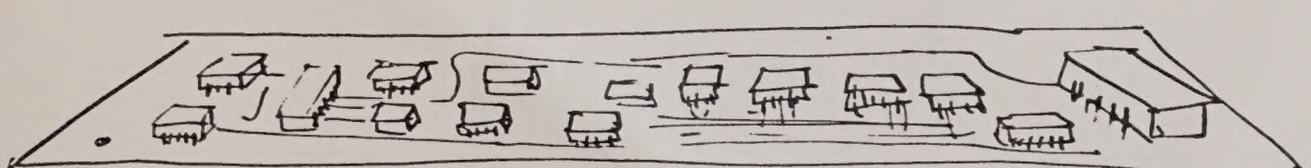
EFFECTIVE DATA STRUCTURE: TABLES, STACKS, MARKERS, POINTERS

ELEMENTARY DATA STRUCTURES: bytes, fragments, chunks, words

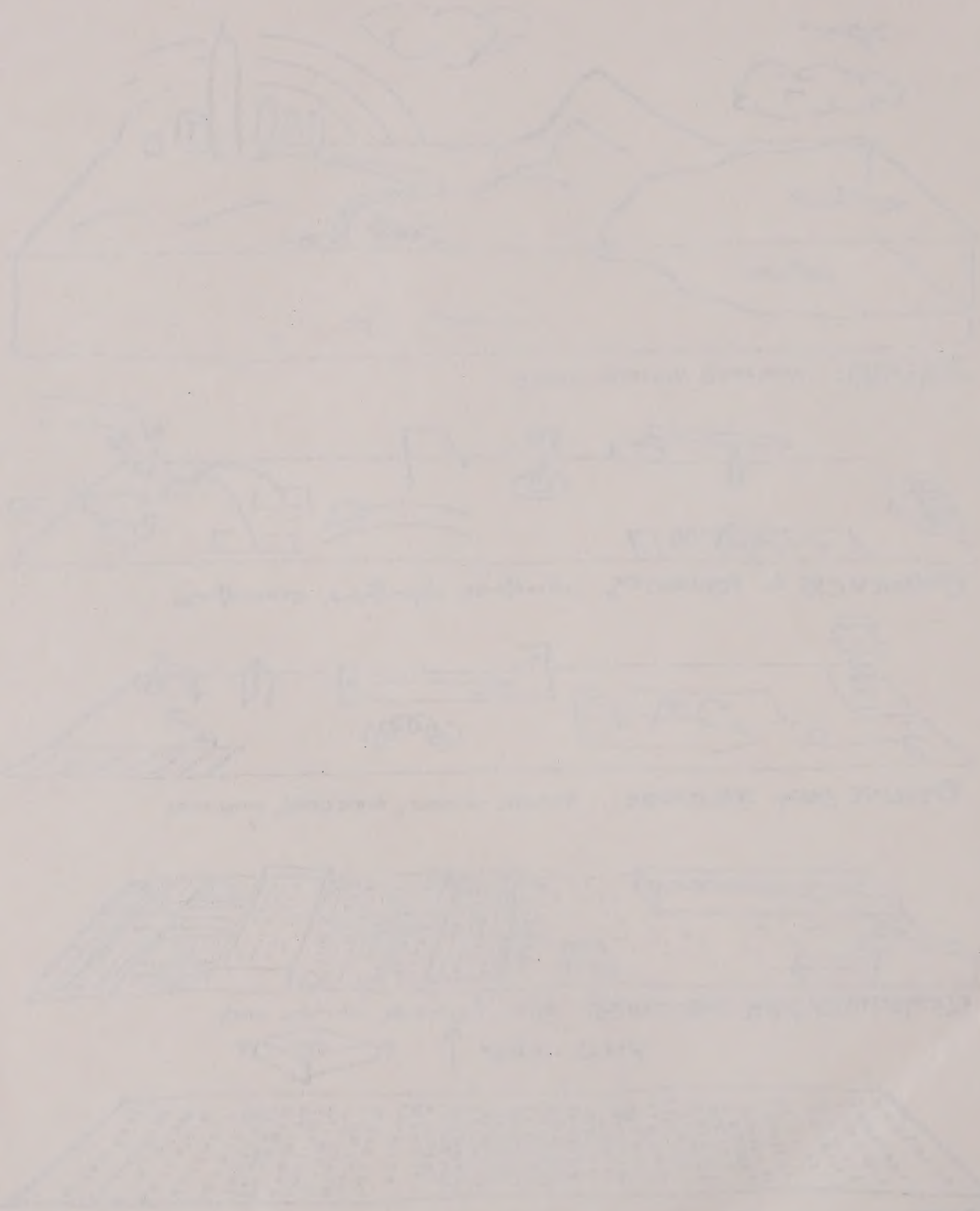
WALLS ARISE



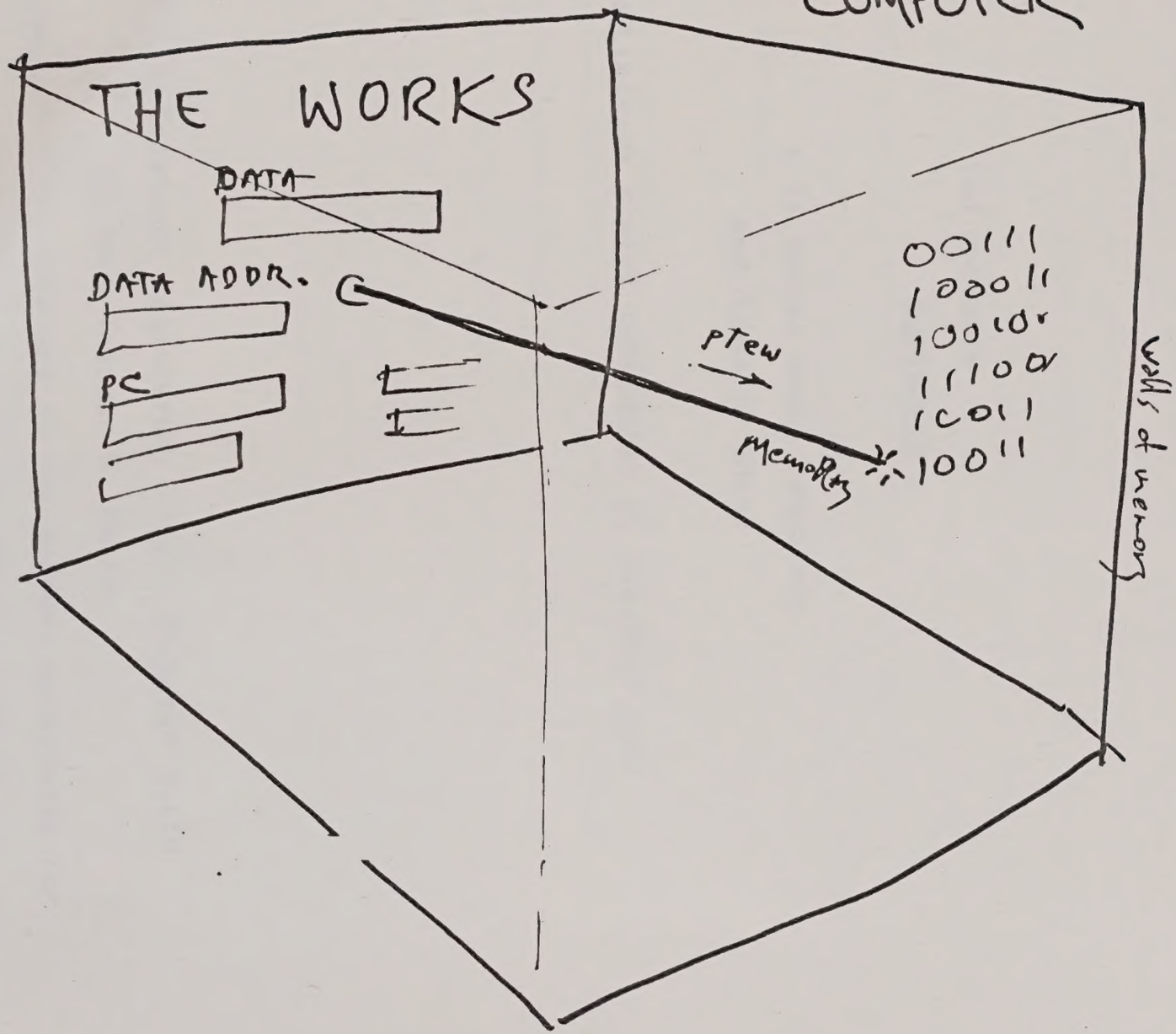
BITS



HARDWARE (chip layout & traces)



2. THE TRANSPARENT COMPUTER



Digitized by the Internet Archive
in 2025

https://archive.org/details/wectd00tedn

Ted Nelson's

SOFTWORLD *

A TV Series about Computers

and Computer Ideas

PRELIMINARY SCRIPT TREATMENT: 21 December 1982

Title & treatment registered, Screen Writers' Guild

(c) 1982 Theodor H. Nelson

"SoftWorld" is a trademark of the author, originally for his personal computing software package.

Other trademarks cited are also those of the author and will become part of the tie-in package.

This is a proposal for a thirteen-week, half-hour series on computers.

That the subject is vital and of growing importance need hardly be stated. That some people will not want to know about the subject is certain; that thirty million people _will_ want to see it-- the ten million already involved with computers, the twenty million who would like to know more-- is likely.

The idea is to do a show on computers in a format like that of _Civilisation_, _Connections_, _The Body in Question_, _Life on Earth_ and _Cosmos_. The format is simple and familiar: a subject is covered vividly and romantically, in 13 half-hour (actually shorter) segments, by an urbane, confident and knowledgeable narrator who knows the field well and presents it with a uniquely personal point of view.

STRUCTURE OF THE PACKAGE

Anatomically, the thirteen segments must be no more than twenty-four minutes each (being further subdivided into major sections of about twelve, six and six minutes for

possible commercial breaks). In other words, some 300 minutes in all.

The different segments should fit together into a whole; but given the nature of television, they must fit together in a complementary fashion regardless of the order in which they are seen. Yet each must stand alone, and each must contribute to the viewer's understanding.

Second Series. It is important also to have in mind a second round: given the great marketability of such a series, more than a little thought should be given to a second thirteen weeks, on the assumption that it will probably be wanted eventually. Leaving plenty of play in the material for additonal exposition and visuals is therefore important.

COMPETITIVE POSITION

A series "on computers" is something obvious to do, and somebody is going to do it. We may not be the first; and that's okay. This show will be unique in style and content, and its unique viewpoint will be deeply grounded in knowledge of the field.

PRODUCT TIE-INS

Shows like Cosmos and Life on Earth can only sell a book. The possible tie-in sales for a series like this, by contrast, are staggering. Books of course, like other series we've mentioned; but each of these created just one book, where ours could spawn several, including especially introductions to various popular computers and languages as discussed in the series. But note also:

Software

There is an extraordinary opportunity for selling **software**, not only to individuals and families, but to schools from elementary through university level. They could be packaged for Radio Shack, Apple and Commodore computers, as well as any others that make comparable market penetration.

Languages. Processors for the major computer languages explained on the show-- especially Logo, Pascal, C, Forth, Cobol, Fortran. PL/I, APL, LISP and Smalltalk-- could be packaged as a set. (These are already available individually, but the marketing power of tying to our show would allow them to be marketed for less than they go for now; I am sure their developers would relate well to such a proposition.) Another very desirable possibility, however, would be commissioning modified mini-versions especially for

teaching purposes. (We do not, of course, want to encourage the learning of Cobol, Fortran or PL/I.)

Hacks-- that is, exercises and things to try-- could be offered for the languages, with good books and audio cassettes-- to enhance the package.

Graphics from the show. The 3D graphics from the show -- animations, traversible jungle gyms of ideas-- will exist as data structures that can be rotated, played with on the screen, and explored to reveal further details and pieces of explanatory text. Adapted for home machines, these graphical data structures, and their manipulative software, will be a sales natural.

Hypertexts. Explorable texts that relate to the show and can be explored on the screen are another natural.

These by no means exhaust the possibilities of software tie-ins. In any case, the important first step is to make the series; products can be put together in a follow-on production stage. (The Star Wars products came out some time after the movie, too.)

Videotapes and videodiscs of the show; spinoff tapes. While the home market is presumably small, schools would want it; and indeed the tapes could be offered in "long

form," with more explanatory detail, longer interviews with key computer people, and additional study material.

PRODUCTION PLAN

Given preliminary financing, the first steps are to rough out the scripts, design the graphical software, and do a preliminary version on videotape. (This preliminary version would be strictly talk, with dummied-up props, and should under no circumstances be shown to anyone who is overly swayed by flossy production values.) The purpose is to have a sort of living storyboard that test sequences can be dropped into, and a context in which any completed piece can be seen.

Then, given the resources, a single pilot with the computer graphics. This should get the money for the full show.

The next step in full production is location work. We want a lot of interview material, particularly with the Narrator present, though this is not strictly necessary (as in network news). (See section on Interviews.)

Meantime, full graphics implementation. Finally, full studio shoot; and creative editing. Because the accidents

form," with more explanatory detail, longer interviews with key computer people, and additional study material.

PRODUCTION PLAN

Given preliminary financing, the first steps are to round out the script, design the graphical software, and do a preliminary version on videotape. (This preliminary version would be strictly bare, with down-and-dirty props, and should under no circumstances be shown to anyone who is overly swayed by flimsy production values.) The purpose is to have a sort of living storyboard that test sequences can be dropped into, and a context in which any completed piece can be seen.

Then, given the resources, a single pilot with the computer graphics. This should get the money for the full show.

The next step in full production is location work. We want a lot of interview material, particularly with the narrator present, though this is not strictly necessary (as in network news). (See section on Interviews.)

Meantime, full graphics implementation. Finally, full studio shoot and creative editing. Because the acclimate

of the interviews wiill have considerable prominence, final editing should use the working script as a point of departure and leave a lot of flexibility for the editing stage.

(One piece of needed development software should be mentioned. It would be very desirable to put together a videotape cataloguing system that would allow automatic re-assembly of new or alternative versions of the show automatically. For this the purchase of controllable VTRs and development of appropriate programs ought to be an actual part of the budget.)

CONTENT

The basic format is time-tested and true: that of the Clever Knowledgeable Narrator strutting through a world he knows like a duke on his grounds, showing you now this, now that.

However, because the subject is so immense, there is plenty of room for travel, location stuff both indoors and out, and many interviews. These should not be merely with "experts," but also with laymen, children, victims, welfare recipients, venture capitalists, dullards and fools, very bad programmers and pompous asses, and very nifty people.

"Experts" should be included with care, and only because their personal interest-- charm, brilliance or perhaps obnoxiousness-- makes them appropriate. Among the finer and more camera-worthy people to be included should be Minsky, Negroponte, Alan Kay, Chris Crawford, Gordon Bell, Jobs and Wozniak of Apple. "Bad guys" from IBM installations and the CIA can probably be found and encouraged to present their views in a way everyone can understand.

The Slant.

Every episode will stress the software angle of its subject. There will be no fixed sequence, but throughout the show we will consider such questions as: What were the early approaches? What are the main software approaches now? What main problems have had to be solved? What modules, techniques, human interfaces were required? Who likes what languages for this? (Always good for controversy, especially with interviews.) Related hardware questions will also be covered, such as: What sorts of computers have they used and do they now?-- but only if the answer has special meaning.

Multithreading

A key aspect of the structure of the show will be the multithreading of topics. That is, while each show will have a specific topic unto itself, care will be taken to mention many different things that are discussed on the other shows, indicating (briefly and by hints) how they tie in. Thus in principle the viewer could continue to find new insights in the second and third viewing. (The experience of my other writings has proven the power and validity of this approach.)

Novelty material.

High-caliber wit and cabaret oddities should be thrown in where workable. I think, for instance, of a tap-dance by Gordon Pask to indicate certain algorithms; of Lee Felsenstein's robot imitations; and of Danny Hillis' genre computer jokes, including variants on ape jokes, sick jokes and Zen jokes.

I consider it also desirable that the show contain something new for everyone: it should offer insights to both the beginner and the expert. (While this may sound difficult, it is an approach I have taken successfully in numerous past writings.)

CONTENT AND PERSONAL STYLE

There are very few people who could do a show like this-- that is, who could participate in the writing and production and serve as host with confident knowledge and understanding of every word and why it is the _right_ word. The combination required, of technical knowledge and judgment, expository skill, wit, savoir-faire, showmanship, stage presence and brass-balled _chutzpah_ is comparatively rare in any field, let alone computers.

However, I know of several other candidates, and so part of the construction of this series is based on the competitive assumption that _there is another such series, and we are programming against it_. Thus the present grand design assumes that some other group with a good narrator and a conventional approach has made just such another series. Therefore this series is designed to complement any other plausible series; and, if need be, stand against a "series on computers" presented from a conventional point of view.

What is the conventional point of view? Something like this: the computer is a glorified calculator; gradually _other uses_ have been found for it; those who use and control the computer are objective scientists, basing what they do on firm knowledge; businesses that use computers generally

know what they're doing; people who program computers generally know what they're doing; all this is based on the firm and steady progress of scientific minds carefully and studiously pushing forward the frontiers. We should trust these brave scientists.

By contrast, the point of view presented here is: computers are not mathematical but general-purpose, though it took us years to realize it; there is nothing remotely like "objectivity" in the computer field, but rather numerous sects and cults passionately pushing their own points of view; businesses do things wrong and stupidly all the time; programmers do things wrong and stupidly all the time; rather than progress being based on "objective" criteria, the way ideas move through the field has much more in common with fads and religious conversion. And he who trusts computer people deserves whatever he gets.

The style of thinking might also be characterized as "high software chauvinist." By that I mean the point of view that considers software more important than hardware, and especially that considers <u>well-designed</u> software, elegantly crafted in today's highest languages (such as C and Smalltalk) what computers are really about. The history of computerdom has been a halting progress in breaking things into parts that are both imaginative and manageable. The adversaries, if there are any, are the "chipmonks" who

overemphasize the particulars of equipment; the "bit-fiddlers" who tunnel obsessively in unimportant details that could be better handled automatically; and the "kludgers" who would rather hurry and program things one piece at a time, regardless of the overall result, than go slowly, design thoughtfully, and do things right.

But even though the show has a certain snob viewpoint, that doesn't mean that we look down on video games. So-called "video games," never mind the stupid misnomer, are at the forefront of today's most advanced computer system design.

THE CENTRAL IDEA

Essentially the central idea of "Softworld" is this: that tomorrow's world is to be designed and decided. The future is soft-- a soft world-- malleable, clay in our hands; and we must not be fooled by the guys who say it's technically determined and we should trust them to go get us this destiny they're hustling. We have many choices. However, the choices can probably only be made on a piecemeal basis, and learning to design things this way is one of the lessons we get from good computer practice.

overemphasize the particulars of equipment; the "pit-fiddlers" who tunnel obsessively in unimportant details that could be better handled automatically; and the "kludgers" who would rather hurry and program things one piece at a time, regardless of the overall result, when go slowly, design thoughtfully, and do things right.

But even though the show has a certain shop viewpoint, that doesn't mean that we look down on video games. So-called "video games," never mind the stupid misnomer, are at the forefront of today's most advanced computer system design.

THE CENTRAL IDEA

Essentially the central idea of "Softworld" is this: that tomorrow's world is to be designed and decided. The future is soft-- a soft world-- malleable, clay in our hands; and we must not be fooled by the guys who say it's technically determined and we should trust them to do get us this destiny they're hustling. We have many choices. However, the choices can probably only be made on a piecemeal basis, and learning to design things this way is one of the lessons we get from good computer practice.

THE SNOWBALL

I am amazed and delighted at the momentum this project has already gotten with minimum effort: at the people on both coasts (and in between) who have expressed their enthusiasm and desire to contribute; at the general agreement that the show should have a firm point of view, rather than being an anthology to please everybody; and at the considerable encouragement I have received along this project's short path so far. If things go on like this it might just be Gang Busters all the way.

THE GRAPHIC STRUCTURES/ORNAMENTS

Central to the show are a number of 3D graphic constructions, to be made with care at a topnotch 3D computer graphics house. They all appear as "ornaments" in the opening titles, and all usually figure in each episode; we get to know them well visually, and the enthusiastic viewer may purchase them as explorable, animatable structures for his home computer: the Carpet of Data, the Transparent Computer, SimLand, the GrafBox, the Tree of Machines, the Tree of Languages.

The graphic structures are colorful and full of motion, and each appears in different versions depending on context. The structures also marry to each other, especially the first three.

Transitions between structures-- showing them changing-- has important explanatory value, especially as we go from level to level of the Carpet of Data, watching the data take on different manifestations.

Scale: the apparent sizes of the graphic structures change on demand from small and hand-sized to room-sized or

larger, with Narrator holding one in his palm or walking about in or on it. The Narrator may slide on the data like ice, tap-dance on it, or walk around simulated machinery in action, coming around the other side with new props in hand.

THE CARPET OF DATA

The Carpet of Data shows the many ways of mapping data into storage. It is actually a multi-level carpet, or even a club sandwich.

(Separate illustration 1.)

The first level of the Carpet of Data is just ones and zeroes-- divided into words or bytes or not divided at all. (The ones and zeroes may be random or revealed to have meaning.)

In the second level, the Carpet of Data is divided into elementary structures or significant segments. Different parts are different colors to indicate different meanings-- numbers, text, programs, tables-- but each segment overlays the ones and zeroes that had been there before. (Thus they are not interpreted, say, as ASCII, since that would involve

larger, with Narrator holding one in his palm or walking about in or on it. The Narrator may slide on the data like ice, tap-dance on it, or walk around simulated machinery in action, coming around the other side with new probs in hand.

THE CARPET OF DATA

The Carpet of Data shows the many ways of mapping data into storage. It is actually a multi-level carpet, or even a club sandwich.

(Separate illustration 1.)

The first level of the Carpet of Data is just ones and zeroes-- divided into words or bytes or not divided at all. (The ones and zeroes may be random or revealed to have meaning.)

In the second level, the Carpet of Data is divided into elementary structures or significant segments. Different parts are different colors to indicate different meanings-- numbers, text, programs, tables-- but each segment overlays the ones and zeroes that had been there before. (Thus they are not interpreted, say, as ASCII, since that would involve

re-spacing them.)

The next level up we may call the Effective data structures. Here the data are partitioned into segments, and the segments are connected by pointers and have frames around them; these are the data structures as they are seen in the imagination of the programmer. This presentation can be quite creative, extending into the third dimension: arrow-spinners, little cranes, little catapults may be used. (This level will of course be used in the explanation of data structures, as well as elsewhere.)

(The lower levels of the Carpet of Data-- the bit, word and data-segment levels-- wrap around the Transparent Computer as ways to visualize its memory.)

SimLand (below) is the uppermost level of the Carpet of Data.

THE TRANSPARENT COMPUTER(tm)

This is a picture of the insides of a computer. It's roughly a cube, though it takes on different proportions depending on the context and the machine whose structure is being shown. (It may even become trapezoidal.) One side shows operating registers and some other inexplicit details,

called WORKS. The other sides are called MEMORY, and show the different slots of RAM or core, or various mappings of the Carpet of Data.

(Separate illustration 2.)

The Transparent Computer has different manifestations, showing the different particular structure of different famous mahines. These changes include the number of visible registers, length of registers, length of memory words.

The Memory consists of three walls and the floor. Their size and divisions depend on the machine, and on what we are calling attention to. The Carpet of Data, wrapped around these four sides, shows us the memory merely as bits, or divided into bytes, or divided into words of the machine's particular size, or partitioned into grander data structures. The amount of memory seen-- effectively whether the bits are small or infinitesimal-- depends on whether we are dealing with a big machine or small.

The Works is the fourth wall. It includes operating registers (depending on the machine being shown), an instruction register, a memory location register and whatever seems appropriate for the explanatory points being

(Separate illustration 2.)

made.

The Works also include a clock (labelled TIMER). This has major cycles, used (for instance) to clock memory store and fetch on particular instructions. To indicate register operations, we see operands actually changing in the main registers as the timer shows its phases.

The MemoRay(tm)

To indicate memory-reference operations, a visible ray-- like the weapons in Star Wars, perhaps even with the ptew-ptew sound-- darts out from the Works to the appropriate location in Memory. The contents can be seen to jump in the appropriate direction.

For the showing of bus machines (such as the PDP-11), the bus is also shown as a multichannel path.

Input and output paths lead upward from the Works, out of the box, and over to other transparent devices. These include a Terminal, which can also be viewed transparently.

Note that dual-processor arrangements (including subroutining displays and other DMA devices) may be simulated, using a doubled wall of Works and two memory rays.

SIMLAND(tm)

SimLand is a landscape with mountains and valleys, rivers and forests, cities, refineries, oil wells pumping, traffic on freeways, space-ships about to take off, planets whizzing overhead, armies on the march (especially tanks), airplanes flying over, bombs going off. There is as much of it as we care to make.

(I would like the design-- esp. of cities and machines-- to be done in 1930s Futuristic Modern Deco; for this I would particularly like to get sketches from Bruce McCall, the National Lampoon artist/writer.)

(See separate illustration 1:
top level of "The Carpet of Data")

SimLand is, of course, a 3D solid-model world. Realism can vary, since it's supposed to show the general idea of simulation (which certainly varies in precision).

SimLand can be reached as the top layer of the Carpet of Data-- but you can get to it by other routes. Simland even sometimes appears in the memory of the Transparent

Computer-- but that must be handled carefully to avoid confusion.

GRAFBOX(tm)

GrafBox is the package for explaining and simulating three-dimensional graphics. It consists of:

A "simulated scene"-- consisting first of polygons, later of curved patches;

A "perspective box"-- explained to be either hardware or software;

Rays indicating projection, and a virtual lens that they pass through.

A projected view, in pixels. The projection is of course valid for the changing simulated scene, and the simulated pixels may of course be magnified.

(Addable for later explanations:)

A "front surface method"-- twanging its way through the scene data, registering corresponding changes on the projected view.

Without the front-surface method, the picture going to the simulated screen is a wire-frame view. Addition of the front-surface method makes it a shaded scene simulation. (Shadows are not for the first thirteen weeks.)

Naturally, this should all be programmed re-entrantly, so that the Grafbox can show the three-dimensional projection of any of the 3D graphics on the show, including itself.

THE TREE OF MACHINES

This is a colorful 3D arrangement showing the different families of computer architecture. It is a tree in the computer sense, meaning that it hangs down rather than growing up; at a distance it looks like a sprig of mistletoe, except for the colors and the names hanging in space, and little 3D models of what the machines look like.

(It is not a true tree, because the link structure also shows cross-influences.)

For each major machine, we can dive into its interior, which is of course the animated computer interior again-- as varied for the particular machine.

THE TREE OF LANGUAGES

This is a colorful 3D arrangement showing the different

branches of computer languages. It is a tree in the computer sense, meaning that it hangs down rather than growing up; at a distance it looks like a sprig of mistletoe, except for the colors and the names hanging in space. (It is not a true tree, however, since some of the connections show cross-influences.)

In its usual view we see only the names of the main common languages, especially Assembler, Autocoder, RPG, Cobol, Fortran, C, Basic, PL/I, Ada, Forth, Lisp, Logo, APL, Smalltalk, Snobol, Constraints. However, on magnification we sometimes pull in to see how many more there are. (We will use the 1000 names of computer languages from Grace Hopper's ca. 1975 book, plus all the others we can find.)

For each major class of language, there will be **animations of typical operations**-- subroutine, string substitution, etc.-- and typical data structures in action (arrays, linked lists...)

SOME INDIVIDUAL EPISODES

Here is a list of one possible season of shows. What is presented here is to be read for content and flavor; nothing is fixed. The software emphasis doesn't show in these synopses, nor do the tie-ins and little repeated explanations that allow the audience to start anywhere. I have placed the more generalized introductory material at the end to stress that these can be in any sequence.

FREEDOM

What 1984 was really about. Quotes from other Orwell stuff. Conjectures about impersonal and computerized societies. Clips from the films "1984," "THX 1138," Star Trek episodes, Nazi documentaries and propaganda.

The FBI. The CIA. The NSA. Proposed bills to limit personal freedom.

THE BILL OF RIGHTS. The Virginia original. Its stormy history. Packwood's proposed amendment for electronic freedom of speech and the press.

THE RIGHT TO PRIVACY, if any. The highly-suspect DES code, approved by the government. Trapdoor codes, Rivest code, the Inman letter. Censorship of papers on coding.

SAVING OUR FREEDOMS. (What if it comes down heavy?) Samizdat publishing in Russia and its possible electronic equivalent under tyranny.

VIDEO GAMES.

They're computers; why aren't they called "computer games"?

Arcade video games and their architecture. (The Zgrass computer is a converted Gorf arcade box.)

HOW THEY'RE PROGRAMMED. A&B-buffer animation, as in Donkey Kong and Pac-Man.

Vectoring, as in Tempest.

HOME VIDEO-GAME MACHINES. Very tight architecture of the Atari VCS; the incredible contortions it takes to program it. The "Frob" development system; the rivals who program for the VCS.

FUTURE OF VIDEO GAMES: nothing less than the future of the human mind: graphics programming as the key to tomorrow's offices and library access.

THE HOT NEAR FUTURE. Architecture of the Evans and Sutherland machines. What Atari is doing under Alan Kay, even though Alan Kay won't tell.

MOVIE TRICKS BY COMPUTER

(Basic computer concepts told through everybody's

favorite subject. We play games in the telling with Spielberg-style clouds and eerie music.)

Movie basics. The strip of film and its transformations (an example of mathematical group theory). Running time backward. Old clips from Melies showing double exposure, matte painting, travelling matte, perspective tricks, pyrotechnics.

CUTTING TRICKS. Editing systems that speed up editing and show more combinations in less time (CMX, Cinenym).

MONSTERS AND CRITTERS: Puppetoons and Go-motion. (King Kong, Speedy Alka-Seltzer, E.T. Also gila monsters with glued-on parts.) DIGITAL RECORDING OF MOTIONS.

Path-recording and motion-control cameras-- just like Go-Motion, but you move a camera instead of a monster. HOW TO MAKE A SPACE FLEET; pivotal position of the librarian who keeps track of all the strips of film; automating <u>that</u> job.

SCENE SYNTHESIS IN 3D-- samples. (That's for a separate episode.)

NUCLEAR WARS

Simulations of What Might Happen; in a sense, we've been continuously in nuclear war since 1955.

Computerization of Command and Control and reliable communications. Cheyenne Mountain, the Looking Glass Plane.

Interview with a former SAC pilot ("hell, I had a South Sea Island all picked out." Interviews with Kahn,

Caldecott, Schelling.

Various endgames. (In our SimLand we see: missile silos; Polaris submarines under the sea; then spectacular visuals simulating different wars and outcomes.)

(Have a nuclear-political simulation game that can be played at home?)

COMPUTER MUSIC SYNTHESIS

A marvelous area for teaching basic computer concepts. This show is heavy on the sound track.

We start by looking at how video games and Apples make music, then going on to the hard stuff.

Internally: a data structure problem with multiple levels of interpretation. Internal data structures of music-generating systems. Conventional music notation as terrible notational engineering.

Waveforms as output subroutines. BANDWIDTH OUTPUT PROBLEM.

Orchestrating your Apple with keyboards. (Salesmen for Alpha Syntauri and Keyboard Chaser explain their competing programs.)

Professional musical experimenters, from Laurie Spiegel's Apple work to Bill Buxton's CRT-controlled music lab. Future effects of (and with) digital recording.

COMPUTERS IN SPACE

Avionics: from the Spruce Goose to the Shuttle and 787. The "adjustable ride" of the F-111; from the centralized computers of the Shuttle to the dispersed processors of the new 787.

Data base for space: keeping track of objects in space (already a problem). Knowing what trash and colonies will be where when (tricky indexing).

Astronautics: finding and feeling your orbits and paths. Anatomy of orbits. Getting around the solar system.

Space colonization: what society will there be? L5 interviews. The dark scenario of prison ships, company towns, military bases, earth as hostage. The happier scenario: what O'Neill's homesteaders would need in a barter-economy asteroid belt.

Libraries in orbit?

THE NOTATIONAL ENGINE (computer architecture)

The **box that follows a plan** is the fundamental idea. In our Transparent Computer we see plans followed in various ways.

Computer languages are then the different notations for setting forth these plans.

Trying to say what you mean: how good languages and data make all the difference. Made-up example: the problem of piano-movers communicating-- Chaplin, Laurel & Hardy

COMPUTERS IN SPACE

Avionics: from the Spruce Goose to the Shuttle and 737. The "adjustable ride" of the F-111; from the centralized computers of the Shuttle to the dispersed processors of the new 737.

Data base for space: keeping track of objects in space (already a problem). Knowing what trash and colonies will be where when (tricky indexing).

Astronautics: finding and feeling your orbits and paths. Anatomy of orbits. Getting around the solar system.

Space colonization: what society will there be? Lo interviews. The dark scenario of prison ships, company towns, military bases, earth as hostage. The happier scenario: what O'Neill's homesteaders would need in a barter-economy asteroid belt.

Libraries in orbit?

THE NOTATIONAL ENGINE (computer architecture)

The box that follows a plan is the fundamental idea. In our Transparent Computer we see plans followed in various ways.

Computer languages are then the different notations for setting forth these plans.

Trying to say what you mean: how good languages and data make all the difference. Made-up example: the problem of piano-movers communicating-- Chaplin, Laurel & Hardy

clips.

The structure of the suroutine and what it took to understand it. "The Stack giveth, the Stack taketh away." Parentheses and why the match the structure of the stack and of the subroutine. Lukasiewic and "Polish notation."

Theory of notation: say it well, understand it better. Ways notation affects thinking and doing; interaction between notation and what you can do.

SOFTWARE IN HARD DISGUISE

A software world increasingly surrounds us. More and more things we buy or meet have software as a central part, yet masquerade as "machines": cash registers, home video games, CAT scanners, reading machines, sewing machines, orchestral music synthesizers and play-along-with-you electric organs.

How people react, why they think they're buying or dealing with "equipment." Most people don't want to hear the truth about it-- it's too vague.

Sometimes very dumb programs get in the box: case histories. And, as usual, you can't hook up your computerized sewing machine to your personal computer because they didn't allow it. How equipment can be tied together-- maybe, sometimes.

The shift of emphasis from hardware ("how do you build it?" to software ("How can we change it later?")

SPELLS AND PRAYERS (program structures and languages)

All programming is casting spells on-- or, if you will, offering prayers to-- a machine set up to carry them out.

The problem is how to ask for what you want. And every computing language is a different approach to that problem, from another point of view, someone else's idea of a good way to give directions.

Forget about "objectivity:" it's the most emotional issue in the field, highly sectarian, closely analogous to different religious denominations and sects.

Religious wars of computer languages and "structured programming." Tony Hoare's famous claim: "The ADA language will start World War III." Interviews with adherents. Different enactments of the same programs in Basic, Fortran, Pascal, Snobol, Logo, Lisp, Smalltalk, APL, Forth.

Problems of space and time, multiple entities and multiple processors. Why real time, simulation, ultradependable systems and video games are different ballparks-- and why they aren't.

Higher and higher language: more clarity, more freedom, less trivia but also less machine efficiency. How to fix.

DOING IT TO ITSELF (subroutining and recursion)

Subroutines. **Graphical sprites.**

The stack. (The Burroughs 5000 and the PDP-11, the two classical stack machines.) The FORTH stack language, enacted by tap-dancers.

RECURSION. Hofstader examples and jokes. LISP and LOGO; why children can write recursive programs even though adults can't understand the concept.

GRAPHICAL EXAMPLES: "time tunnel" effects, self-similar zooms.

Fractals explained. Piece of fractal mountain movie, McKenna's easier-to-understand fractal pictures.

CRUNCH (evolution of computers)

Using our Transparent Computer, with many variations, we look at the changing and growing computers, and how history keeps repeating itself with divided machines, two-headed machines, and two-level machines.

To avoid being dry, this show has an increasing carnival atmosphere. Each new machine, a variant on our Transparent Computer, gets added to a carnival midway with escalating carousel and calliope music; Narrator becomes a barker for each-- no dull exposition. Crowds of extras press forward, want them all.

The first machine: Babbage and Lady Lovelace.

The divided machine: separation of instructions and data. Eniac; Eckert, Mauchly and the group (all still alive but Mauchly).

The two-headed machine: two separate program followers, one memory. DMA devices as a second computer. Subroutining displays, vector processors. Sutherland's "great wheel of reincarnation." (Multiple heads.)

The two-level machine: microcode. Simulating one machine with another one but faster. The pseudo-11 they had to stop selling (interview with Omahundro, its designer). Carrying this to absurdity: the IBM 1500 simulating a 360 simulating an APL machine.

Big classical machines. The PDP-6 and its bastard son, the 360. Today's two-level 370s.

Bus machines and the rebirth of multi-headed architecture.

(By now the bus architecture has become a roller coaster. A whole Coney Island has been created. Camera pulls away. It's SimLand, getting smaller and smaller.)

THE COMPUTER WORLD

Programmers. Travel agents. Embezzlers. Executives. Chipmonks. Venture capitalists. Head-hunters.

An innocent at her First Programming Job. Bosses trying to boss what they don't understand. People buying what they don't understand on the basis of buzzwords they don't understand.

Elites. Grunts. The dark Satanic Cobol mills. The holies of holies (AILab, SAIL, PARC, Watson Lab).

The two-headed machine: two separate program followers, one memory. DMA devices as a second computer. Subroutining displays, vector processors. Sutherland's "great wheel of reincarnation." (Multiple heads.)

The two-level machine: microcode. Simulating one machine with another one but faster. The pseudo-11 they had to stop selling (interview with Omahundro, its designer). Carrying this to absurdity: the IBM 1500 simulating a 360 simulating an APL machine.

Big classical machines. The KDF-6 and its bastard son, the 360. Today's two-level 370s.

Bus machines and the rewirth of multi-headed architecture.

(By now the bus architecture has become a roller coaster. A whole Coney Island has been created. Camera pulls away. It's Simland, getting smaller and smaller.)

THE COMPUTER WORLD

Programmers. Travel agents. Embezzlers. Executives. Chipmonks. Venture capitalists. Head-hunters.

An innocent at her First Programming Job. Bosses trying to boss what they don't understand. People buying what they don't understand on the basis of buzzwords they don't understand.

Elites. Grunts. The dark Satanic Cobol mills. The holies of holies (AiLab, SAIL, PARC, Watson Lab).

Establishment straights. Cottage software crazies.

Kids, kids, kids, programming in schools, sneaking onto big systems.

Where is it going? (We show that programming schools are preparing people for obsolete work.) Adam Osborne: "Out of Control."

2D GRAPHICS.

Simplest form: characters on a terminal (real and shown on the terminal of our Transparent Computer). Fancier versions: overlays. Animation.

Subroutines explained through Pac-Man.

Sprites in Logo.

Examples of 2D graphic software available for home computers.

Optical scanning for input. The Digisector for the Apple; the fancy Israeli scanners.

Output for printing. Output for movies. (Old Knowlton clip.)

2D automata languages (e.g. BEFLIX, Conway game of Life.)

SHADES OF REALITY (3D graphics)

What's available for home computers. What's been seen in the movies already. (Peter Fonda mask in Eutureworld,

III juggler, Star Trek II "Genesis" sequence, Tron.) Csuri, Japanese "living shape" animation, Whitted's glass balls, Blinn's Saturn flyby, Carla's Island (show closeups of the clouds and sunset). "Sunstone," The Works.

Evans and Sutherland equipment. The already-legendary PIXAR computer being built at Lucasfilms.

How it's actually done, shown with our 3D graphical simulations. The old GE simulator, the Warnock method, Watkins boxes and today's dispersed methods.

A visit to Digital Effects Studios, New York.

SO WHAT IF YOU MADE ME? (Artificial Intelligence.)

Chess: teaching the computer what fork to use. Heuristics. Lisp. The "AI Techniques." The issue of Consciousness.

LIVING UP TO THE ROBOTIC MYTH. Clips from "Metropolis," "The Day the Earth Stood Still," "2001," "Dark Star." Dirty-talking phony "robots" at conventions.

Minsky and the gang at AILab, MIT. The opposition: Dreyfus and Weizenbaum. What do the kids think? What can you buy now?

YOUR COMPUTER HOME

How you could set up a home life-- a rather strange home life-- just with what's on the market. Dick Don's

robot home north of Chicago; the "Xanadu home of the future" at Disney world.

Other visions of unified computerization of home and personal life, with immense control-- yet none of the trappings of today's computerish style.

NEW WAYS OF SEEING

Animating a multidimensional cube.

Baecker's new analytic geometry of color.

Path movements. Molecular simulations in movies. Interactive diagrams.

Imagined cockpits in "2001," "Alien," "Star Trek."

CAT scans, ultrasound scans, dynamic spatial reconstruction of the beating heart.

Feeling molecules as you reposition them.

What is the sound of one quark snapping?

These new ways of seeing are arbitrary new inventions-- all a part of tomorrow's informational sensorium. And some you can put on your home computer.

INTERVIEWS

Many brilliant and delightful people there are in the computer world, and most of them ought to be available for interviews. (Indeed, most of the _founders_ of the computer field are still with us, with only a few lamented exceptions, and there is still time to get them on tape.)

Consequently, a large part of the production should consist of interviews with computer people-- both Greats and Nifties. However, the whole series is only about three hundred minutes, and probably no more than one-sixth should be interview snips, if that much.

Therefore the interview materials would be cut three ways: into the main show; into the second season; and into spinoff tapes of the interviews themselves (possibly organized around other topics).

In other words: we should go to lengths to get extensive interview material, enough for a talk-show with each interviewee; use in the series particular _bon mots_ and little touches revealing the interviewees' character; and archive the rest for the inevitable demand that will come

later. (Comparable experience with talking records made in the fifties has shown great anguish over the loss of the out-takes; we can do better.)

Cost of road-work. Obviously, this approach hinges on the cost of production. If we treat this like a full-scale Hollywood or TV-commercial operation, touring the country on such interviews would be absolutely prohibitively. But if we can consider that acceptable interview-quality materials could be gotten with the lighter videotape equipment (such as Technicolor), or with Super 8 film, then the cost of shooting interviews is not much more than the cost of sending two people around the country, which is what it would take. (Personally, given the quality of video, I hold by Super 8.)

THE PEOPLE

There follows a list of people we should consider talking to, with a capsule summary of why I think they would be interesting.

(I am personally acquainted with 46 of the following people, which may be helpful.)

LEN ADELMAN. (Co-developer of Rivest-Shamir-Adelman code. See Rivest.)

AMDAHL. Designer of IBM's 360 (along with Blaauw and Brooks), he and his son have gone on to found the major 360-compatible competitor firms.

DAVE AHL. Founder of Creative Computing, he got there when the getting was good and plugged away. Knows very little about computers and the magazine is called "Flake City," but he's managed to get his.

ATANASOFF. Got some sort of patent on computers in the nineteen-forties and was thus declared the main inventor by a Minnesota court; is laughed at by those who built Eniac.

GORDON BELL. Rumpled genius who designed the PDP-8 and PDP-11 computers; director of research for Digital Equipment Corp.

JOHN BACKUS. The inventor of FORTRAN; co-inventor of Backus-Naur recursive notation; now an emeritus designer of computing languages and computing language theory.

RON BAECKER. First to work to work in computer animation; inventor of the moving-path curve.

BLAAUW. Designer (along with Amdahl and Brooks) of IBM's 360 computer, he is especially noted for the invention of paging hardware.

JIM BLINN. Star programmer of 3D graphics movies for Jet Propulsion Lab and creator of all their simulation movies; handsome, clever, good speaker.

DAN BRICKLIN. Inventor of Visi-Calc; instant multimillionaire.

FRED BROOKS. Designer (along with Amdahl and Blaauw) of IBM's 360 computer, he then went on to lead the appalling development of OS/360, the world's most dreadful operating system. He has fully repented, offering as penance his book The Mythical Man-Month. As chairman of computer science at the University of North Carolina, he has built a computer-graphic facility for use by visiting chemists who want to explore molecular structure and folding properties visually.

NOLAN BUSHNELL. Founder of Atari, which he reputedly sold for seventeen million; more recently, the founder of Chuck E. Cheese's Pizza Time Theater.

BILL BUXTON. Canadian, has an automatic music synthesis system controlled almost entirely from CRTs.

RENEE CHIPMAN. Lovely, brilliant and incredibly spacey lady of the MIT crowd. "Did she really say that?" is most people's open-mouthed reaction to her whimsical generalist remarks. Voted "Miss Information of 1982" at Hypercon 8.

PAT COLE, very attractive young woman who did the comic-strip computer-graphic history of the world for Cosmos; now head of special projects for Atari.

DIFFIE. (With Hellman, first proposed trapdoor codes. See Rivest.)

JOHN DRAPER. As the world-famous "Captain Crunch," is the only person to have served time for Phone Freaking. Now, as "Captain Software," is doing nicely off EasyWriter. Some

would call him a loon.

HUBERT DREYFUS. With Weizenbaum, a leader of the anti-artificial-intelligence camp. Formerly of Rand.

JOHN DUNN. Young retired programmer who lives in gorgeous setting in California. (Success story.)

J. PRESPER ECKERT. Co-creator of the ENIAC (with John Mauchly); still vice-president of Univac.

LEE FELSENSTEIN. Designer of the SOL and Osborne computers, clean-cut Lee is a shirt-and-tie Marxist; also gives very funny "robot" imitations.

MARK FOWLER. Current head of the FCC, and a surprise spokesman for freedom in electronic media.

ADELE GOLDBERG. Breathless redhead with a whim of iron; shepherded the Smalltalk project and Alan Kay for a decade at Xerox PARC; uses her anger as a weapon throughout the field.

WAYNE GREEN. Unpopular publisher of magazines and software; originally had a monopoly on amateur radio magazines. Founded Byte magazine in his wife's name (see Virginia Londner), a move he has since greatly regretted. Now publishes others.

DICK HEISER. Mellow Californian; founder of the world's first computer store.

CARL HELMERS. "Founding Editor" of Byte and (in my opinion) a complete idiot. (See Londner.)

NAT HENTOFF. New York writer; spokesman for free speech and bill of rights.

HELLMAN. With Diffie, first proposed trapdoor codes. (See Rivest.)

DANNY HILLIS. Minsky's right-hand whiz kid even when he was an MIT freshman; also a brilliant stand-up comic and the author of the best computer jokes in the field.

DOUGLAS HOFSTADER. Poet of recursion (author of Goedel, Escher, Bach). Successor to Martin Gardner's "Mathematical Games" column in the Scientific American, which he has renamed "Metamagical Themas."

GRACE HOPPER. Little old lady and naval commander, she created COBOL and forced it on the Department of Defense--and, indirectly, the business community.

BOBBIE INMAN. One of the spooksmen for the forces of darkness and evil. Involved with the infamous "NSA letter" to suppress the RSA code (see Rivest). Later, as head of the CIA, proposed that all scientific publication should first be cleared with government security.

KENNETH IVERSON. Inventor of APL; crisp, clever, intolerant.

STEVEN JOBS (of Jobs 'n' Woz). With Wozniak, founded Apple Computer. The Business Head of the two.

ALAN KAY. Inventor of Smalltalk language and for a decade the resident star of Xerox Palo Alto Research Center; now Chief Scientist of Atari, with a reputed $100 million annual budget. Witty, mischievous but earnest.

SCOTT KIM. Famous artist of visual calligraphic pun; also very sensitive individual, grad student in computer science.

KLUDGE. The first editor of Datamation, his name playfully was made the synonym for "ridiculous computer or hookup."

KEN KNOWLTON. One of the pioneers of computer graphics, he made practically the first computer movies, worked out the first graphic languages and a lot of other good stuff.

DONALD KNUTH. World's chief chronicler of algorithms. Can be playfully embarrassed by citing his first publication, "The MAD System of Weights and Measures."

THOMAS KUHN. Author of The Structure of Scientific Revolutions; himself revolutionized theories of how science evolves. Very smart guy.

VIRGINIA LONDNER. Nominally the publisher of Byte while married to Wayne Green, she took it with her after the divorce. After several marriages since personal computing began, she has reverted to her maiden name. (See Green, Helmers.)

DAVID LUBAR. Under 25, his first job was assistant editor of Creative Computing. Teaching himself programming on his own Apple, he rose to an immensely lucrative position as his second job-- programming games for the Atari VCS at Sirius.

JAMES MARTIN. Thundrous establishment spokesman, also knows a lot.

KAY MAUCHLY. Widow of John, she worked with him and Eckert on the Eniac project as a young woman. Charming and intelligent.

AARON MARCUS. Dignified and grave, a graphic artist working on interactive systems. He is so dignified, in fact, that

he can walk through an airport with a yarmulkah sporting Mickey Mouse ears and still project total serenity.

NELSON MAX. Wispish, effeminate individual, but makes dynamite movies by computer-- especially "Zooms on Self-Similar Shapes," molecular stuff and "Carla's Island."

JOHN McCARTHY. Inventor of LISP language, the AI standard.

BRUCE McDIARMID. Coiner of the term "notational engineering;" has the unique ability to read LISP in normal English, translating as he goes.

DAVID McLEY. Builder of the McLeyvier, an orchestra-in-a-box which merely presents to the user a keyboard and CRT.

DIANA MERRY. Most unusual woman (six feet, deep voice); works at Xerox PARC on the Smalltalk project.

MARGARET MINSKY. Daughter of Marvin; sweet, roly-poly, incredibly nice; currently shepherding LOGO, Inc.

MARVIN MINSKY. The King of Artificial Intelligence; witty, playful, intolerant, charming, incredible.

LYALL MORRILL. His first computer program, "Whatsit?", turned into a business. Wears a helicopter beanie, active in San Francisco gay politics.

NICHOLAS NEGROPONTE. The Great Toymaker. Runs the Architecture Machine Group at MIT, spending millions of dollars in Defense Department Advanced Research funds in making colorful and hilarious toylike objects and environments.

DAVID NITZAN. Roboticist at SRI. Dynamic.

KEN OLSEN. Founded Digital Equipment Corp., building the

first PDP-1 computer in his garage. Still president.

GORDON PASK. Loony eccentric theatrical theorist of artificial intelligence.

SEN. ROBERT PACKWOOD (R., Washington). Staunch defender of the Bill of Rights; has advocated a Constitutional amendment for freedom of electronic information.

SEYMOUR PAPERT. Luminary of the computer counterculture; creator of the LOGO language, especially good for teaching math to kids.

HOWARD PEARLMUTTER. Gabby FORTH programmer.

ROSS PEROT. The "instant billionaire" whose software company grew to staggering size by promising to take care of your IBM installation with no hassle. (Also staged his own commando raid in Iran to rescue his employees there.)

HARRY PYLE. Created the first Datapoint computer, which later became the Intel 8008-- the predecessor of all of today's chip computers.

RON RESCH. World expert on large-scale design of buildings by folded sheets; has astounding movies of his unique origami; created immense Easter egg sculpture by computer for a small Canadian village on a grant from the Mounties.

RONALD RIVEST. Principal developer of Rivest-Shamir-Adelman code, one of the central protections of freedom and privacy in the world of tomorrow; the best so far of the Trapdoor Codes.

ED ROBERTS. As head of MITS, an almost-bankrupt firm making radio controls for model aircraft, Roberts in desperation

advertised a personal-computer kit for $400 and inadvertently started the personal-computing tidal wave. Seizing the moment, he piloted MITS to solvency and retired. (And on the way, he designed the S-100 bus, now an industry standard.)

TODD RUNDGREN. Rock star and visionary who's done professional software development for the Apple.

RAVI SHAMIR. (Co-developer of Rivest-Shamir-Adelman code. See Rivest.)

LAURIE SPIEGEL. Thoughtful, sweet and very intellectual New York City computer musician, artist, writer. (Her loft is a trip and should be shown.)

GERALD SUSSMAN. Creator of the Constraints programming language and a brilliant thinker at the current edge.

IVAN SUTHERLAND. The George Washington of computer graphics, he created the original Sketchpad at MIT (clips of which we can find). His clarity of vision and sense of elegant design far overshadow those of most others in the field, and most of the people in the field are damn good. He went on to found Evans and Sutherland Computer Company, the Rolls-Royce of computer graphics.

JOSEPH WEIZENBAUM. Better known than Dreyfus, he is one of the leaders of the anti-AI faction, and the most hated by the AI people. Like the top AI people, he is at MIT.

JOHN WHITNEY. Founder of computer art movies, beginning with his rebuilding of WWII bomber equipment; invented slit-scan; now does abstract visual ballets of dots.

STEVEN WOZNIAK (of Jobs 'n' Woz). The technical half of the founders of Apple (with Steven Jobs). A former phone freak, Wozniak used to sell blue boxes on the Stanford campus. After designing his computer, he asked his employers, Hewlett-Packard, if they wanted to claim it. They said no. He named it Apple because he was at the time a Fruitarian. His 7% of Apple is supposedly worth over $200 million. He sponsored the "Us" rock festival this year.

LEOR ZOLMAN. Lovable young MIT dropout who has made a bundle selling his compiler for the C language to Z-80 hackers.

CLASSES OF PEOPLE.

We also want interview one-liners from lots of other people-- programmers, passersby, welfare recipients. Also and especially:

Kids.

Children who program and care about computers, in all different styles: California; snooty private schools; nursery school; parochial school.

Women in computing.

There are lots of smart ones; let's hear from them.

Chipmonks.

The obsessed people who think it's all hardware will not get equal time. They should, however, get a chance to express their views-- pithily.